Extracted from:

Explore Software Defined Radio

Use SDR to Receive Satellite Images and Space Signals

The Pragmatic Bookshelf

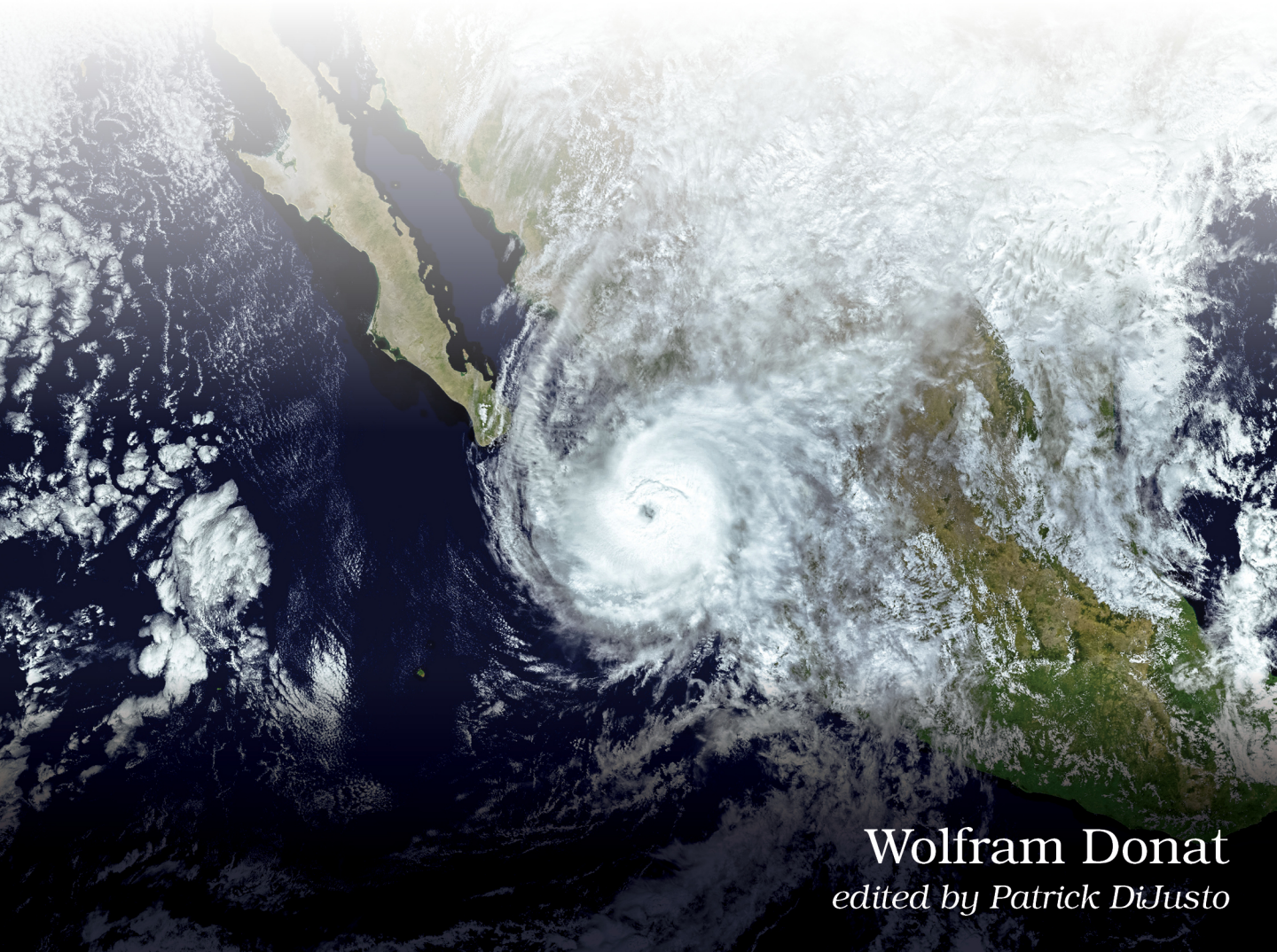
Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic exPress

Explore Software Defined Radio

Use SDR to Receive Satellite Images and Space Signals



Wolfram Donat

edited by Patrick DiJusto

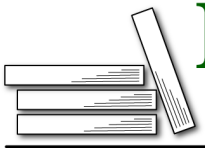
Explore Software Defined Radio

Use SDR to Receive Satellite Images and Space Signals

Wolfram Donat

The Pragmatic Bookshelf

Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Executive Editor: Dave Rankin
Development Editor: Patrick DiJusto
Copy Editor: L. Sakhi MacMillan
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

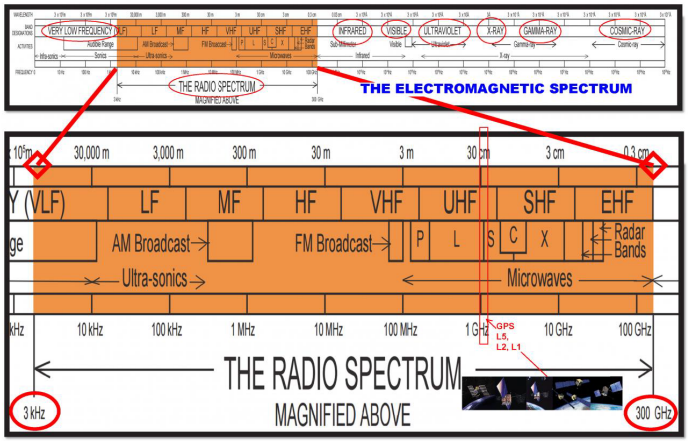
For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-759-1
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—February 2021

So you've decided to explore the wonderful world of software-defined radio, the world of virtual transistors, analog-to-digital and digital-to-analog converters, and homemade antennas. For that, I say congratulations and welcome! I also say good luck and Godspeed, for here be dragons. I have buckled my armor and girded my loins for the express purpose of guiding you through the turbulent waters of turning your computer into a radio receiver. *And if that isn't a mishmash of metaphors, I don't know what is. Anyway...*

With the proper SDR tools, including software, tuning device, and antenna, you can use your computer to tune into a large swath of the radio spectrum, from 64 MHz—the lower part of the VHF bands—all the way up to the 1700 MHz UHF bands and beyond. There are tools (Airspy HF+, for instance) that will tune into the spectrum all the way down to 1 KHz, and still others, such as HackRF and LimeSDR, that will go higher than 1700 MHz. These are a bit on the expensive side and perhaps beyond the scope of this introductory book, but know that they do exist should you want to explore more of the fringes of the radio spectrum. Instead of turning a tuning dial to change radio stations, as you do with your car radio, with SDR you merely tell the software to tune the device to a specific frequency. This can be done with astonishing levels of precision; some software will allow you to increment or decrement your scan in units of 0.00001 MHz. That's well beyond what even the most precise, steady-handed person can achieve with a manual tuner. As you can see in the image from http://www.transportation.gov, SDR doesn't cover a huge portion of the radio spectrum, but having one device able to tune into that many frequencies is very impressive.

Given the right antenna and software, it's often possible to use the same USB dongle to listen to FM radio stations, CB radio (see sidebar), police and fire scanners, the International Space Station, and even to tune into and visualize the signals from NOAA weather satellites.



Once the signal is acquired, it's routed from the antenna, through the tuning hardware (most often a USB dongle of some sort) into your computer's sound card. The sound card acts as a digital-to-analog converter, taking the analog signals and converting them to digital signals which your computer can pipe to the speakers. Some signals, such as those from weather satellites, have an image embedded into the signal being transmitted. With the correct software, you can extract and view those images.

Getting started with software-defined radio can be a challenging experience. The SDR world is still a fledgling area with little documentation, despite numerous subreddits, websites (http://www.rtl-sdr.com, for instance), and online/Facebook groups. As I write this, Hackaday is even hosting an online chat with SDR guru Harold Giddings, who goes by the call sign KR0SIV. SDR may be growing in popularity, but it's still kind of hackerish.

As such, what few SDR software programs exist are often buggy, platform-specific, and—most of all—poorly documented. Dozens of hours of research can lead to nothing but confusion and frustration. When I was getting started, the frustration was often palpable; there's nothing like following a long set of instructions, step by step, for over an hour, only to find when you finish that it doesn't work and it's anyone's guess as to why.

One thing I learned through my travails and experiments is that it's important to be pragmatic when it comes to the tools—including the operating system—you decide to use to play with SDR. I'm primarily a Linux guy, for example; if you've read any of my other works, you'll know that I am most comfortable in Ubuntu and Raspbian. I do, however, use both Mac and Windows as well, as I'm not a purist and will happily switch to whatever tool is

Citizens Band Radio Service

In the United States, Citizen's Band, or CB, is a two-way, short-distance voice communications service operating near 27 MHz in the shortwave band. It can be used for both personal and business messages. It probably reached its height of usage in the 1970s and early 1980s, but the availability of pagers and, later, cell phones have returned the service to its original users, long-haul truck drivers and hobbyists. Younger readers may not be familiar with it, but just watching an old movie like *Smokey and the Bandit* may introduce you to nostalgic phrases like "10-4, good buddy" and "chicken lights" and "There was a plain brown wrapper at the 60-yard stick, a bear in the air, and a wreck at the 405. The coops were workin' hard on your side going west."

The original CB specs called for AM transmission, but over time channels 36 through 40 became used for SSB communication. To listen to CB conversations, tune to one of the MHz frequencies below on AM. From 27.365 MHz and up, use either LSB or USB.

- 26.965 | 26.975 | 26.985 | 27.005
- 27.015 | 27.025 | 27.035 | 27.055
- 27.165 | 27.175 | 27.185 | 27.205
- 27.215 | 27.225 | 27.235 | 27.245
- 27.255 | 27.265 | 27.275 | 27.285
- 27.295 | 27.305 | 27.315 | 27.325
- 27.335 | 27.345 | 27.355 | 27.365
- 27.375 | 27.385 | 27.395 | 27.405

When you're comfortable with the tools discussed in the book, try listening in and see if there's any CB traffic in your local area!

best for the particular job I'm doing. And I almost always try to keep my books OS-independent, giving instructions for all three major OSes.

However, when it comes to SDR, Windows is still the OS of choice, so we'll often be using that when it comes to the projects in the book. Linux definitely has some software out there, and I had great success with some of it, but unfortunately, it seems that many of the Linux tools are very hardware dependent; the same piece of software may work fine on your desktop system, but switch to one with a different USB chip and it all may fail. In addition, some tools are just easier to download and use in Windows than in Linux.

As for you Mac junkies, OSX (or MacOS) now has many of the same tools available that Linux has, including rtl-sdr, airpsy, and hackrf. `GQRX`, the main

Linux tool I use in this book, is also available from either its website or via homebrew or macports. However, while I was writing this book, I stayed firmly in the Linux and Windows arenas, so I don't know how much success you may have with these tools or how easy they may (or may not) be to install. Just know that they exist, and if you try them out and they work well, please let me know!

Hardware

Let's talk about the hardware that's necessary for any software-defined radio experimentation. The first thing you're going to need, obviously, is a radio—or its equivalent in the SDR world: a USB dongle. Most dongles in the SDR space have been originally designed as TV tuners, to allow the user to receive HD TV signals out of the air. As their popularity has grown (for both tuning into television signals and for software-defined radio enthusiasts) they've come down significantly in price, and many of them can be used to detect all sorts of signals, given the right antenna. Repurposing these dongles fits one definition of hacking: making a device do something it wasn't built to do.

The most common SDR dongles you're likely to see use the RTL28xx interface and the Realtek R82xx tuner chipsets, housed in a variety of different packages. The dongle I'm currently using (see the image that follows) is from NooElec and is available from your favorite online retailer for around $20 (https://www.amazon.com/gp/product/B009U7WZCA/).

Others are available, of course, from various retailers, so don't feel any pressure to purchase one over another. That being said, however, remember my earlier warnings about getting different setups to work? If you're completely new to the SDR world, it may behoove you to duplicate my efforts here exactly, starting with the hardware I'm using. I would hate for you to duplicate my steps exactly but have the project not work because of some vague mismatch between your hardware and your software, or because your USB device isn't readable by your SDR software.

The next piece of hardware you'll need is an antenna, to grab all of those beautiful signals out of the air and funnel them into your SDR dongle. Chapter 4 is all about antenna design and theory and which antennas will do the best job for particular projects and signals, but when you're just getting started and getting familiar with the processes involved, you just need any old antenna.

The NooElec dongle in the previous link comes with an antenna; my experience is that the included antenna is worth about as much as a snowblower in the Mojave. Two hours spent trying to listen to the local radio station and failing to get anything convinced me to try another antenna I had picked up when I was still unsure as to what I needed. Don't be afraid to switch antennas, as switching antennas can make all the difference, along with placement (which we'll get into later as well). When you're first getting started and are just trying to pick up some signals—any signals—you may have good luck with this one: https://www.amazon.com/1090Mhz-Antenna-Connector-2-5dbi-Adapter/dp/B013S8B234/ (see the following image). I certainly did. Out of all of the pieces in the SDR puzzle, the antenna may make the most difference. You may get your setup to work perfectly, but if your antenna is wrong (such as being the wrong design or having the wrong placement), you may have no luck picking up signals. Feel free to experiment.

We'll be switching up our antenna for our later projects, but this is a good one to start with. Whichever one you choose, make sure that the connector matches the connector on your dongle, which is most likely an SMA (the first

of the two images that follow) or an MCX (the second image). Happily, many add-on antennas come with an array of adapters to fit most any radio device.

You will most likely want to get a longer cable. The stand-alone antenna I bought, for instance, comes with a 1-meter cable, which is plenty for some simple experimentation—picking up your local radio station, for instance. However, success with SDR depends not only on the antenna but the antenna *placement*. Getting the antenna far away from your computer and other noisy devices is crucial, especially as the strength of the signal you're trying to receive decreases. As I said, we'll get into antenna design a bit later on, but a longer cable is almost guaranteed to be a necessity. Again, make sure the extension cable you choose fits not only the antenna but the connector on your USB dongle. Also make sure your genders are correct on each end of the cable; you may want to purchase a selection of gender-changing adapters to go with your SDR toolbox.

That's the bare minimum of hardware you'll need to start experimenting. Read on for an introduction to the software we'll be using.

Software

Ah, the software. Here's where things can get a bit sticky, so bear with me while I attempt to lead you through the jungle. In a nutshell, you'll be finding

and installing a new device driver and a tuning program. Sounds simple, right?

Using your USB dongle for SDR experiments requires, at a minimum, a device driver that is more adaptable than the standard manufacturer's or Windows or Linux drivers. Once you get the new driver(s) installed and working, you'll also need software that enables you to tune the dongle to your choice of frequency. This is often called RTL-SDR software. Windows users most often use a program called SDRSharp, while Linux users tend to use a package called GQRX.

Windows

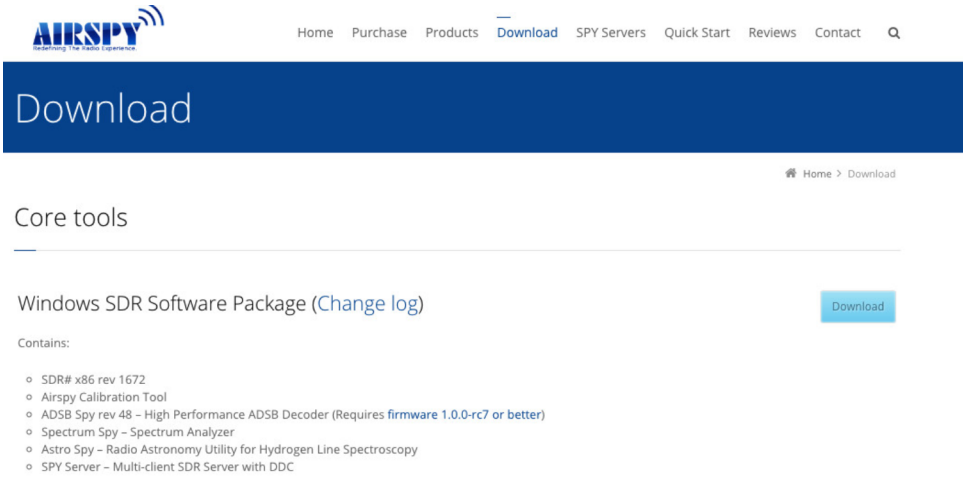
For simplicity's sake, all the Windows work you see here will be done on the latest build of Windows 10. It's *likely*, though not guaranteed, that the packages you see will also work with Windows 7 and 8. A newer computer is also a good idea, but anything as powerful as a dual-processor or better should be fine.

SDRSharp, the most common Windows program, requires Microsoft's .NET version 4.6 or newer to be installed. If you're using Windows 10, this may already be installed, but don't count on it—I tested this process using a brand new install of Windows 10 Home edition and .NET 4.6 was missing. You may also need the Visual C++ runtime. (Don't worry—you won't be doing any C++ or .NET programming; those libraries are just necessary to compile and run the software.)

To install .NET, go to https://www.microsoft.com/en-us/download/details.aspx?id=55167 and choose your language from the pull-down menu. Click the big red Download button and follow the instructions to install it. The Visual C++ package is very similar; go to https://www.microsoft.com/en-us/download/details.aspx?id=8328 and again choose your language. Click the Download button and follow the instructions to install the package.

When the installation is finished, you should now have all of the operating system tools you'll need. Now, point your browser to https://airspy.com/download. Click the Download button next to Windows SDR Software Package (as shown in the image on page 8).

When the download is finished, you'll have a file named sdrsharp.zip in your Downloads folder. Move that file to a directory of your choosing and extract it by right-clicking and selecting Extract Here. You'll end up with a new directory called sdrsharp-x86. (Don't worry if you're running a 64-bit system; the software will install and run just fine.)



Navigate inside that directory and run install-rtlsdr.bat. Make sure you're connected to the Internet before you run the .bat file. A command prompt should open, which will then attempt to download two new files to your directory: rtlsdr.dll and zadig.exe. *Both* of these files are necessary to proceed. The file rtlsdr.dll is the modified driver for your SDR dongle, and zadig.exe is a handy tool for telling Windows to use the new driver instead of the old one (because the old one won't work for SDR experiments).

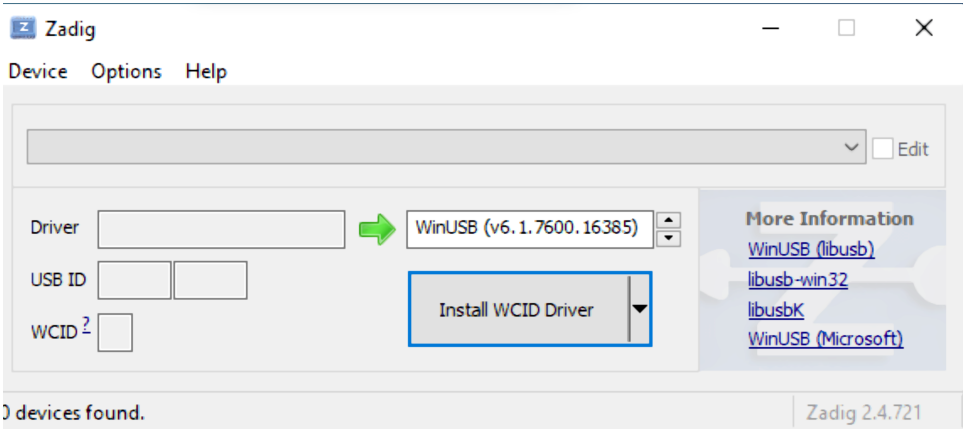
Although zadig.exe downloaded for me without any problems, rtlsdr.dll did not. If this happens to you, you'll need to download that file manually. Point your browser to http://osmocom.org/attachments/download/2242/RelWithDebInfo.zip, which will put a new RelWithDebInfo.zip file in your Downloads folder. Unzip the file the same way you did the sdrsharp.zip folder, and you should have a new directory named rtl-sdr-release.

Navigate inside that folder to either the x32 or x64 directory, depending on what version of Windows you're running (most likely 64-bit). Copy this new rtlsdr.dll file into the sdrsharp-x86 directory.

If by chance the zadig.exe file didn't download correctly, enter https://github.com/pbatard/libwdi/releases/download/b730/zadig-2.5.exe into your browser window and copy the resulting .exe file into your sdrsharp-x86 directory.

That finishes off the software you need to download. Now you need to run everything. Plug in your dongle and wait for Windows to try to find or install drivers for it. Don't worry whether it succeeds or not, since you'll be replacing those drivers in a moment.

When it's finished, open your sdrsharp-x86 directory, right-click the zadig.exe file, and select Run as Administrator. This will open the following window:



Zadig is a nifty little tool that lets you choose what drivers you want to use for a particular device—in our case, the SDR dongle. In the menu bar, select Options and make sure that there's a check mark next to List All Devices. Also, uncheck Ignore Hubs or Composite Parents to make sure that you can see everything connected to your computer.

Now, in the main drop-down menu, you'll need to select your dongle. It should appear in one of two ways: as Bulk-In, Interface (Interface 0), or as something like RTL2832UHIDIR or RTL2832U. Choose whichever one shows up in your menu. You'll know you have the right device when the USB ID showing is 0BDA 2838 00. Oddly enough, every tuner dongle will have this USB ID, no matter the manufacturer. Do *not* select anything else, because this can severely screw up your USB drivers.

Underneath the drop-down menu, you'll see a box labeled Driver, which is prepopulated with whatever driver Windows happened to select for your dongle. In the box next to it (which the big green arrow is pointing to), make sure WinUSB (vX.X.XXXX.XXXXX) is selected. This is the driver you're going to use to replace the Windows default. Click the big blue Replace Driver button.

You're almost guaranteed to get a warning about unverified publishers and unsigned drivers; just ignore it and install the software anyway.

You should now have the drivers necessary for your SDR dongle to work. It's possible that if you unplug your dongle or move it to another USB port, you *may* need to run zadig.exe again, so don't delete it from your sdrsharp folder.

On to the next chapter for your first radio reception.